بايثون بلمسة

معدلة

محتويات الكتاب:

1- عن الكتاب:

أقدم لكم هذه الكتاب المتواضع لتعليم لغة برمجة البايثون وقد حاولت قدر المستطاع أن يكون بسيط وخصوصا للمبتدئين الذين يجدون صعوبة في البدء بالتعلم.

ويتناول هذا الكتاب المواضيع الأساسية في لغة برمجة البايثون 3 مع الأمثلة التوضيحية.

ملاحظة:

- **جميع الأكواد في الكتاب مجربة وعلى الاصدار بايثون 3 .**
- **في حال وجود اي سؤال أو استفسار الرجاء مراسلتي.**

كتب في سوريا – دمشق بتاريخ 2012/7/27

2- عن الكاتب:

مصطفى فرحات مبرمج ومطور تطبيقات مهتم بعدة لغات برمجة وخصوصا البايثون ،مؤسس موقع بايثون بالعربي وهو أول موقع مختص بلغة برمجة البايثون، من هواياتي التدوين ومشاركة المعلومات مع الاخرين

مدونتي الرسمية

حسابي على الفيسبوك وتويتر



3- الكتاب برعاية:



www.ar-python.com

4- مقدمة:

- لغة البايثون (Python Language):



هي لغة متعددة الأغراض وغرضية التوجة (OOP) ومن اللغات العالية المستوى.

أهم ما يميز لغة برمجة البايثون أنها سهلة للتعلم وخصوصا للمبتدئين حيث يجد متعلموها سهولة في كتابة وقراءة الشيفرات وتحريرها.

الميزات الأساسية لهذه اللغة:

1- تعمل على الكثير من المنصات (Windows,Linux,Mac).

2- قابلة للتوسع والتطوير.

3- تدعم الواجهات الرسومية (GUI Programming).

4- دعم التعامل مع غالبية قواعد البيانات.

ملاحظة :

- البايثون لغة حساسة لحالة الأحرف مثلا: PROGRAM تختلف عن program.

البدء مع البايثون:

- لتحميل اخر نسخة، الاطلاع على أحدث الأخبار، أو تحميل الكود المصدري قم بزيارة الموقع الرسمي للغة برمجة البايثون.

الموقع الرسمي | Python Official Website

- لتحميل التوثيق (Documentation) وهو متاح بالواحق Html , Pdf

Python Documenation

5- التحضير لبيئة العمل:

سوف نبدأ بالمهم وهو تحضير بيئة العمل ، حيث أنه لا يمككنا البدء بكتابة البرامج بهذه اللغة بدون تحضير بيئة عمل مناسبة للمبرمج تكون متناسبة مع احتياجاته وان تتميز بسلاسة, فأننا نعلم ان التطوير على سكربت او برنامج ليس بالأمر السهل دائما وسيصبح أصعب عند عدم توافر البيئة المناسبة للتطوير.

-يمكن استخدام البايثون بأحد الطرق التالية:

1- Python Shell ويتم عبرها تنفيذ الأوامر سطر تلو اخر .

2- IDLE GUI : كتابة سكربت بايثون بطريقة متقدمة أكثر وتنفيذها.

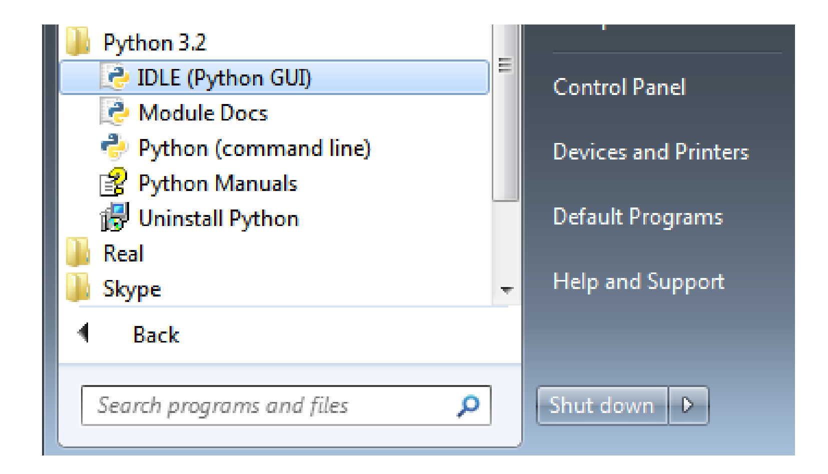
3- Text Editor : نستخدم فيها اي محرر نصوص يعمل على النظام وحفظ الملفات باللاحقة py.

اختيار الـ IDE المناسب:

1- PyScripter: مناسب جدا اذا كنت تستخدم نظام ويندوز
2- Eclipse: محرر الشهير مع اضافة PyDev
3- Netbeans تعمل على الويندوز ، لينوكس،ماك
4- python tools for visual studio : اذا كنت تستخدم الفيجوال ستديو فيمكنك تحميل هذه الاضافة لتضيف ميزة دعم البايثون في الفيجوال ستديو.
5- Eric IDE: ممتاز اذا كنت من مستخدمي لينوكس.

6- كتابة البرنامج الاول بلغة البايثون:

بعد تصيب البايثون نذهب الى القائمة ابدا ثم كافة البرامج وننقر على البرنامج التالي:



وبذلك تكون جاهز لكتابة أكواد البايثون

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

7- البرنامج الأول (برنامج الترحيب):

نبدأ بكتابة أول برنامج لنا بلغة برمجة البايثون ، وهو عبارة عن برنامج يطبع عبارة (Hello word) وتكون الشيفرة بالشكل التالي:

```
print('Hello word')
```

ويكون ناتج التنفيذ

```
Hello word
```

8- التعليقات:

كل ما يكتب بعد اشارة # يعتبر تعليق وسوف يقوم المفسر بتجاهله تماما

مثال:

```
# This is the first list
x=2

# Second line

m=5
```

كما يمكننا كتابة اكواد البايثون بسطر واحد على ان تفصل بين كل تعليمة والتي تليها فاصلة منقوطة ;

مثال:

```
x = 'foo'; print(x);
```

9- الكلمات المحجوزة في البايثون:

كما في معظم لغات البرمجة ، يوجد كلمات محجوزة لا يمكن استخدامها وهي فقط محجوزة للغة البرمجة البايثون، يمكننا معرفة الكلمات المحجوزة بسهولة بواسطة الكود التالي:

```python
import sys
import keyword


print ("Python version: ",
sys.version_info)

print ("Python keywords: ",
keyword.kwlist)
```

فيتم طباعة اصدار البايثون مع الكلمات المحجوزة في اللغة

```
Python version:  (3, 2, 'final', 0)

Python keywords:  ['and', 'assert',
'break', 'class', 'continue', 'def',
'del', 'elif', 'else', 'except',
'exec', 'finally', 'for', 'from',
'global', 'if', 'import', 'in', 'is',
'lambda', 'not', 'or', 'pass',
'print', 'raise', 'return', 'try',
'while', 'yield']
```

10- الأرقام والعمليات الحسابية:

التعامل مع الأرقام والعمليات الأساسية هي نفسها الموجودة في الالة الحاسبة ، وهذا مثال شامل عن (الضرب، القسمة،الطرح،الجمع).

```
print(2+2)

print(4-2)

print (18/3)

print(18/7)

print(18.5/6.7)

print(9%4)

print(8%4)

print(8.75%5)

print(6*6)

print(7*7*7)

print(8**3)

print(-10**3)
```

ناتج التنفيذ يكون كالتالي:

```
4
2
6.0
2.5714285714285716
2.761194029850746
1
0
3.75
36
343
512
-1000
```

11- المتحولات:

كما في لغات البرمجة الأخرى مثل C++,Java, Visual Basic البايثون تحتوي الأنواع الأساسية للمتحولات حيث يمكن أن تكون رقمية (Integer) أو رقمية بفواصل (Float) أو نصية (String) .

```
# رقمية متحولات
x=15
x+10
print(x)
y=4
print(x+y)
#بفاصلة رقمية متحولات
m=1.5
n=3.5
print(m+n)

#نصية متحولات
h="me"
l="you"
print(h)
print(l)
```

ويكون الناتج كالتالي:

```
15
19
5.0
me
you
```

كما انه يتم التصريح عن متحولات عاملة مسبوقة بالكلمة Global مثال:

```
global e

e=10

print(e)
```

12- الـ Modules و Function :

تقسم لغة البرمجة بايثون الى عدة Module حيث تحتوي كل واحدة على تنفيذ عدة وظائف معينة مثال:

Math : مسؤولة عن التعامل مع الارقام والعمليات الرياضية.

- **ويكون الاستخدام على الشكل التالي:**

Module.function

مثال:

math.sqrt

- **يتم استدعاء الـ Module بالعبارة Import.**

مثال:

```
print(pow(3,4))

print(abs(-12))

print(abs(6))


#--------------

import math

print(math.floor(17.6))

print(math.sqrt(81))

#Module.function
```

كما يمكن للاختصار التصريح عن متحول واسناد له الوظيفة المطلوبة.

مثال:

```
#Use variable

t=math.sqrt

print(t(9))
```

13- المحارف (String):

كل ما يكتب بين ".." أو '...' يتعامل البايثون معها على انها معطيات نصية .

مثال:

```
a="Ali "
b='Mohammed'
print(a)
print(b)
print(a+b)
print("HI 'Moustafa' ")
```

ويكون الناتج كالتالي:

```
Ali
Mohammed
Ali Mohammed
HI 'Moustafa'
```

ولطباعة متحول رقمي مع نص نستخدم الكود التالي:

حيث استخدمنا تابع التحويل لنص ()Str

```
mm=str(44)

print("the number is " + mm)
```

فتظهر النتيجة:

```
the number is 44
```

مثال شامل:

```
str = 'Hello World!'

print (str)          # طباعتها كلمة
print (str[0])       # طباعة أول حرف
print (str[2:5])     # الطباعة بدءا من
print (str[2:])      #  الطباعة بالبدء
من
print (str * 2)      # طباعتها مرتان
print (str + "TEST") # طباعة الكلمتين
```

ويكون الخرج كالتالي:

```
Hello World!
H
llo
llo World!
Hello World!Hello World!
Hello World!TEST
```

14- أنواع البيانات في البايثون:

لغة البايثون تحتوي على 5 أنواع قياسية للبيانات وهم:

1- الأرقام

2- النصوص

3- القائمة | List

4- Tuple

5- القواميس | Dictionary

1-القائمة | List:

القائمة تحتوي على عدة عناصر يفصل بينها بفاصلة ومغلقة بالرمز []

مثال على انشاء قائمتين بالبايثون حيث تلاحظ انه من الممكن ان تحتوي على عدة أنواع من البيانات.

```
list = [ 'abcd', 745 , 2.23,
'Moustafa', 70.2 ]

smalllist = [123, 'Ali']

print (list)           # طباعة القائمة
```

```
print (list[0])        #  طباعة العنصر
الأول
print (list[1:3])      # الطباعة من
print (list[2:])       # الطباعة من
print (smalllist * 2)  # طباعتها مرتان
print (list + smalllist)  طباعتهما مع
بعضهما البعض
```

ناتج التنفيذ يكون كالتالي:

```
['abcd', 745, 2.23, 'Moustafa',
70.200000000000003]
abcd
[745, 2.23]
[2.23, 'Moustafa', 70.200000000000003]
[123, 'Ali', 123, 'Ali']
['abcd', 745, 2.23, 'Moustafa',
70.200000000000003, 123, 'Ali']
```

:Tuples -2

الـ Tuples مشابهة للقائمة ولكن الفرق الوحيد هو انها للقراءة فقط اي لا يمكن اضافة عناصر جديدة بعد انشائها.

```
tuple = ( 'abcd', 786 , 2.23,
'Moustafa', 70.2  )
tinytuple = (123, 'Ali')

print (tuple)           #  طباعة جميع
القيم
print (tuple[0])        #  طباعة العنصر
الأول
print (tuple[1:3])      #  طباعة
العناصر من
print (tuple[2:])       #  طباعة بالبدء
من
print (tinytuple * 2)   # طباعتها متان
print (tuple + tinytuple) #  طباعتهما
مع بعضهما
```

```
('abcd', 786, 2.23, 'Moustafa',
70.200000000000003)
abcd
(786, 2.23)
(2.23, 'Moustafa', 70.200000000000003)
(123, 'Ali', 123, 'Ali')
('abcd', 786, 2.23, 'Moustafa',
70.200000000000003, 123, 'Ali')
```

3- القواميس | Dictionary:

القواميس مشابهة للقائمة ولكن الفرق أنها تحتوي مفتاح-قيمة.

```
dict = {'ali': 'john','code':6734,
'dept': 'sales'}
print (dict['ali'])       # طبع القيمة
المحددة
print (dict['code'])      # طبع القيمة
المحددة
print (dict)             #  طباعتها
كاملة
print (dict.keys())    #  طباعة جميع
المفاتيح
print (dict.values()) #  طباعة جميع
القيم
```

```
john
6734
{'dept': 'sales', 'code': 6734, 'ali':
'john'}
['dept', 'code', 'ali']
['sales', 6734, 'john']
```

15- العبارة الشرطية if :

وهي لاختبار حالة معينة اذا كانت صحيحة فسوف ينفذ مجموعة من الأوامر.

مثال:

```
x=4

if x==4:
    print(x)
```

يكون الناتج

```
4
```

16- العبارة الشرطية elif – else :

- تستخدم else عندما نريد تنفيذ مجموعة أوامر عنا لا يتحقق الشرط أي يكون False
- تستخدم elif عندما نريد اختبار أكثر من حالة للشرط.

مثال:

```
x=3

o=6

y=5

if x==4:
    print(x)
elif o==6:
    print(o)
else:
    print(y)
```

```
6
```

يكون الخرج 6 لان الشرط الثاني محقق وتمت طباعة المتحول O أما لو لم يتحقق الشرطان لتمت طباعة المتحول Y.

17- حلقة while :

طالما أن الشرط محقق سوف تكرر تنفيذ التعليمات التي تليها.

```
i = 4
while i < 9:
    print(i)
    i = i+2
```

سوف يتم طباعة المتحول x ثلاثة مرات الى ان يصبح المتحول I أكبر من الـ 9 ويتم الخروج من الحلقة

```
4
6
8
```

ملاحظة:

يجب الانتباه الى الحلقات اللا نهائية حيث أنه طالما الشرط صحيح سيتم التنفيذ بعدد لا نهائي.

18- حلقة For:

ايضا تعتبر نوع اخر من الحلقات التكرارية في البايثون وهذا مثال على استخدامها:

```python
for letter in 'Python':
    print(letter)
```

في هذه مثال سوف يتم طباعة كل حرف في كلمة Python



```
P
y
t
h
o
n
```

19- عبارة break :

تستخدم للمقاطعة في الحلقات التكرارية For,While

```
for letter in 'Python':
   if letter == 'h':
      break
   print ('Current Letter :', letter)
```

ويكون الخرج كالتالي لانه البرنامج سوف يتوقف عند الوصول الى الحرف H

```
Current Letter : P
Current Letter : y
Current Letter : t
```

20- عبارة continue :

تستخدم عبارة Continue لاعادتنا الى بداية الحلقة وتجاهل باقي التعليمات ، كما أنها يمكن أن تستخدم في For , While معا.

```
for letter in 'Python':
   if letter == 'h':
      continue
   print ('Current Letter :', letter)
```

في هذا المثال سيتم طباعة كل حرف في كلمة Python وعند الوصول الى المحرف H سيتم تجاهل باقي التعليمات والعودة لبداية الحلقة وبذلك يكون الخرج كالتالي

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : o
Current Letter : n
```

ملاحظة:

يمكن استخدام عبارة Else مع حلقات التكرار.

21- عبارة Pass :

تستخدم هذه العبارة عندما لا نريد لأي تعليمات أن تنفذ أي (null operation) ، ولن يحدث اي شيء عند تنفيذها.

```
for letter in 'Python':
   if letter == 'h':
      pass
```

في هذه المثال لن يتم طباعة اي شيء.

22- التصريح عن Function :

يتم التصريح عن الـ Function بالعبارة Def متبوعة باسمها.

مثال يحتوي على Function اسمها myname تحتوي بارامتر str وتقوم بطباعة هذه البارامتر.

استدعائها ببساطة نكتب اسم الـ Function مع البارامتر

```
def myname(str):
    print(str)


#Call function

myname('moustafa')
```

ويكون الخرج كالتالي

```
moustafa
```

مثال اخر على Function تقوم بجمع عددين

```
def num(x,y):
    print(x+y)


num(3,4)
```

ويكون الناتج

```
7
```

23- إنشاء Class:

كما نعلم لغة البايثون لغة غرضية التوجه (oop) وتدعم انشاء الـ Clases

يتم انشاء الـ Class بكتابة كلمة Class ثم اسمه مثال:

```
class example:

    def myname(self,name):
        print(name)
```

هذا مثال على Class اسمه Example ويحتوي على Function اسمها Myname

وتقوم هذه الـ Function بطباعة اسم نحن نحدده.

طريقة استخدام الـ Classسهلة جد وهي بالطريقة التالية:

```
res=example()
res.myname ('Moustafa')
```

حيث res هي الوسيط ويتم الاستدعاء باسم الوسيط متبوعا باسم الـ Function

ويكون الناتج كالتالي:

```
Moustafa
```

مثال 2 :

سنستخدم الـ Function التي استخدمناها في مثالنا السابق والتي تقوم بجمع عددين وسنقوم بانشاء Class ينفذ العمليات الحسابية الأساسية (الجمع،الطرح،الضرب،القسمة)

```
class num1:

    def add(self,x,y):
        print(x+y)

    def sub(self,x,y):
        print(x-y)

    def div(self,x,y):
        print(x/y)

    def mul(self,x,y):
        print(x*y)



result=num1()

result.add (2,2)
result.div (4,2)
result.mul (2,2)
result.sub (4,2)
```

Class اسمه num1 ويحتوي على 4 Functions تمثل العمليات الأساسية.

الوسيط هنا اسمه Result وتم استدعاء الـ Functions واعطاء قيم لارقام.

الناتج يكون كالتالي:

```
4
2
4
2
```

قم بتغيير الأرقام وسيقوم البرنامج بطباعة الناتج.

24- الوراثة Inheritance :

الوراثة في لغة البايثون كما في لغات البرمجة الأخرى هي أن يرث Class بعض خصائص Class اخر .

```
class Parent:
   def myMethod2(self):
      print (' parent ')

class Child(Parent):
   def myMethod(self):
      print (' child ')

c = Child()
c.myMethod2()
c.myMethod()
```

في هذا المثال يوجد Class اسمه Parent و Class اخر اسمه Child

الـ Class الثاني يرث الـ Parent Class

الوسيط هنا هو الـ C وهو للـ child Class

نلاحظ أنه عن طريق الـ Class child تم استدعاء الـ Function mymethod2 من الـ Class parent

يكون ناتج التنفيذ كالتالي:

```
parent
child
```

License

